ΠΟΛΥΤΕΧΝΕΙΟ
ΚΡΗΤΗΣ

ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ

Αναφορά 3ου Milestone Project

Ομάδα: Γαλατιανός Γιώργος 2014030159
Αμπατζίδης Κωνσταντίνος 2014030017

Πρόλογος

Το project είναι μία απλουστευμένη έκδοση του Κινέζικου παιχνιδιού Go και αυτό κληθήκαμε να υλοποιήσουμε. Στο 3ο Milestone μας ζητήθηκε η τελική και ολοκληρωμένη λειτουργία του παιχνιδιού με πλήρη στρατηγική παίχτη και πλήρες πρωτόκολλο επικοινωνίας. Η αναφορά θα είναι διαιρεμένη σε τέσσερα μέρη:

1. Δέσμευση μνήμης και αντίκτυπο αυτής της επιλογής στην εφαρμογή μας
2. Υλοποίηση επικοινωνίας του υπολογιστή με τον AVR
3. Βήματα που υλοποιούν την κίνηση είτε του υπολογιστή είτε του AVR
4. Εξήγηση και περιγραφή του αλγορίθμου της στρατηγικής του AVR

Δέσμευση μνήμης

Η μία επιλογή είναι ο πίνακας να είναι 8x8, δηλαδή 64 θέσεων και ένα byte για ένα κουτάκι της σκακιέρας. Η άλλη επιλογή ήταν να δεσμεύσουμε 1 byte για τέσσερα κουτάκια. Αυτό διότι κάθε κουτί έχει τρεις επιλογές, Black, White, Empty. Με δύο bit μπορούμε να αναπαραστήσουμε όλες τις τιμές, άρα 8bit μπορούν να χωρέσουν την πληροφορία για τέσσερα κουτιά. Αυτή η επιλογή σαφέστατα κάνει καλύτερη διαχείριση της μνήμης. Σε περιπτώσεις που είσαι οριακά αυτό είναι απόλυτα χρήσιμο.

Στα πρώτα milestones μην έχοντας γνώση για την πολυπλοκότητα και το είδος της στρατηγικής που θα εντάξουμε, αποφασίσαμε να χρησιμοποιήσουμε την πρώτη επιλογή για λόγους απλότητας στη σύνθεση και τη δυσκολία του κώδικα. Στο σημείο όμως ένταξης της στρατηγικής είδαμε πως θα ήταν βέλτιστο να είχαμε πράξει αντίθετα. Αναλογιζόμενοι ότι πρόκειται για ένα project εξαμήνου, εμείναμε στην επιλογή μας κατανοώντας ότι θα έχουμε κάποιο άνω όριο στο πόσες κινήσεις θα μπορεί να πάει μπροστά ο αλγόριθμος.

Κάθε κουτί θα περιγράφεται από ένα γράμμα της αλφαβήτου:

- B, για black
- W, για white
- E, για empty

Υλοποίηση Πρωτοκόλλου Επικοινωνίας

Αρχικοποίηση Επικοινωνίας

Στην τελική φάση της εργασίας το baud rate ορίστηκε στα 9600. Αυτό σημαίνει πως με την χρήση του prescaler στα 1024 και του μετρητή 0 ο οποίος είναι 8bit, τα απαραίτητα overflows ανήλθαν στα 39 για να μετρηθεί ένα δευτερόλεπτο. Αυτό υλοποιήθηκε με μια Interrupt Service Routine(ISR) με είσοδο τον πίνακα των overflows. Μόλις είχαμε overflow στον μετρητή, η ISR ενεργοποιούταν και αύξανε τον αντίστοιχο counter. Εντός της συνάρτησης υπήρχε ένας έλεγχος όπου μόλις φτάναμε στο επιθυμητό όριο των overflows τότε μετρούσαμε ένα δευτερόλεπτο.

Αποκωδικοποίηση μηνυμάτων

Σε αυτό το σημείο ακολουθήσαμε την τακτική του τρίτου εργαστηρίου. Με τη χρήση της ISR παίρνουμε τη λέξη που μας καταχωρεί ο χρήστης, την μεταφέρουμε σε πίνακες μεγέθους 15 unsigned char. Έπειτα αποθηκεύουμε τη λέξη και με τη χρήση ενός switch case, ελέγχουμε να είναι εντός των προβλεπόμενων ακολουθιών, ενώ σε αντίθετη περίπτωση εκτυπώνουμε το μήνυμα ER<CR>.

Η συνάρτηση η οποία αναλύει το μήνυμα, επαληθεύει την ορθότητα της δομής του μηνύματος και ξεκινάει την υλοποίηση του, είναι η analyzing_command(). Η συνάρτηση αυτή (μεταξύ πολλών άλλων) μόλις λάβει MV από το PC, τότε επαληθεύει την νομιμότητα της κίνησης του αντιπάλου και εκκινεί την διαδικασία για την απόφαση της κίνησης του AVR. Επίσης εάν λάβει OK αναλόγως με τα flags τα οποία έχουν ενεργοποιηθεί γίνεται η αντίστοιχη λειτουργία.

Σε κάποιες εντολές όπως η RST, η NG αρχικοποιούμε τον χάρτη είτε σε όλα κενά, είτε στην αρχική κατάσταση όπως περιγράφεται στην εκφώνηση, αντίστοιχα. Εντολές όπως οι WN, LS, TE που ανάβουν LED, το ανάβουν κανονικά και παραμένει σε αυτή την κατάσταση έως ότου δώσουμε νέα εντολή.

Υπάρχει αναλυτικό flowchart της συνάρτησης αυτής στον φάκελο της αναφοράς σαν ξεχωριστό αρχείο εικόνας. Λόγω μεγέθους τοποθετήθηκε μόνο του ώστε και με το κατάλληλο ζουμ να μπορεί να είναι ευδιάκριτη η λειτουργία της.

Υλοποίηση κίνησης

Σε αυτό το σημείο χρειάστηκε να δημιουργήσουμε δύο συναρτήσεις, μία για την εύρεση των νόμιμων κινήσεων και μία για την αλλαγή της σκακιέρας.

Εύρεση Νόμιμων Κινήσεων

Η συνάρτηση αυτή έχει ως όρισμα το χρώμα για το οποίο ψάχνουμε τις νόμιμες κινήσεις, αλλά και τον χάρτη στον οποίο επάνω ψάχνει τις κινήσεις αυτές. Επιστρέφει μια struct η οποία μέσα περιέχει μόνο έναν δυσδιάστατο πίνακα με τις κινήσεις. Η χρησιμότητα αυτού θα αναλυθεί στο σημείο της στρατηγικής.

Η σκέψη εδώ είναι πως ψάχνουμε ολόκληρη την σκακιέρα για να βρούμε το δοθέν χρώμα. Μόλις το βρούμε κάνουμε σε οχτώ κατευθύνσεις ψάξιμο για την ακολουθία που πρέπει να ικανοποιείται ώστε να υπάρχει νόμιμη κίνηση. Θα πρέπει στο σημείο που

βρήκαμε το χρώμα, που είχαμε σαν όρισμα, το αμέσως γειτονικό να είναι αντιθέτου χρώματος. Εάν ικανοποιείται αυτή η συνθήκη προχωράμε περαιτέρω στη κατεύθυνση στην οποία έχουμε εισέλθει, όσο βρίσκουμε αντίθετο του δοθέν χρώματος.

Μόλις βρούμε κενό σημείο αυτή θεωρείται μία νόμιμη κίνηση και την καταγράφουμε στον πίνακα που έχουμε δεσμεύσει για αυτό τον σκοπό. Εάν βρούμε χρώμα ίδιο με το δοθέν τότε η αναζήτηση σπάει αφού σε αυτή την κατεύθυνση δεν υπάρχει δυνατή νόμιμη κίνηση. Οι οχτώ κατευθύνσεις είναι οι δύο οριζόντιες, οι δύο κάθετες και οι τέσσερις διαγώνιες. Κατά την ολοκλήρωση της συνάρτησης ο πίνακας Moves θα έχει πλέον όλες τις νόμιμες κινήσεις για το χρώμα το οποίο δόθηκε σαν όρισμα.

Ενημέρωση της σκακιέρας αφού παίχτηκε μία νόμιμη κίνηση

Αυτή η συνάρτηση έχει σαν όρισμα τις συντεταγμένες της κίνησης που έγινε, το χρώμα της κίνησης, αλλά και τον χάρτη πάνω στον οποίο θα γίνει η ενημέρωση. Η λογική είναι αντίστοιχη αφού και πάλι με αρχή το σημείο που έγινε η κίνηση ψάχνουμε προς τις οχτώ κατευθύνσεις για να δούμε που υπάρχει πεδίο για να γίνει η αλλαγή χρώματος. Η ακολουθία πλέον είναι η εξής: Από το σημείο έναρξης προχωράμε στην κατεύθυνση μόνο εάν το γειτονικό είναι αντιθέτου χρώματος και ακολουθείται από μια σειρά αντιθέτου χρώματος αλλά καταλήγει στο χρώμα της κίνησης. Αυτό σημαίνει ότι βρήκαμε ευθεία στην σκακιέρα που στις δύο άκρες έχει το χρώμα της κίνησης και εσωτερικά το αντίθετο, πράγμα που σημαίνει ότι αυτά πρέπει να αντιστραφούν.

Να επισημάνουμε ότι θα πρέπει η μία άκρη της ευθείας να είναι η κίνηση που μόλις έγινε, δεν αλλάζουμε όλες τις ευθείες που ικανοποιούν το παραπάνω. Έτσι στο τέλος στης συνάρτησης αυτής έχουμε την ανανεωμένη σκακιέρα μετά από κάθε κίνηση.

Αυτές λοιπόν είναι οι συναρτήσεις που χρειαστήκαμε ώστε να αρχίσει να παίζει ο υπολογιστής με τον AVR. Η σειρά με την οποία γίνονται όλα είναι προκαθορισμένη και ξεκινάει από το ποιος έχει το μαύρο χρώμα.

Εάν μαύρο έχει ο AVR τότε ξεκινάει τις κινήσεις ο AVR. Αυτό συμβαίνει μέσω της συνάρτησης avrplaying η οποία έχει σαν όρισμα έναν ακέραιο. Όταν αυτός ο ακέραιος έχει την τιμή ένα σημαίνει ότι μπορεί να παίξει κανονικά ο AVR. Υπολογίζει τις κινήσεις που μπορεί να κάνει, εκτελεί την κίνηση η οποία είναι η καλύτερη σύμφωνα με τη στρατηγική που ακολουθούμε και στέλνει το αντίστοιχο μήνυμα στον αντίπαλο με το σημείο που έπαιξε. Η ενημέρωση της σκακιέρας δεν γίνεται σε εκείνο το σημείο αφού πρέπει να λάβουμε την απάντηση ότι όλα έγιναν σωστά. Δηλαδή ότι το μήνυμα στάλθηκε σωστά, και ότι η κίνηση είναι σωστή, το πώς συμβαίνει αυτό θα το πούμε στη συνέχεια.

Εάν ο AVR έχει το άσπρο χρώμα τότε περιμένει κίνηση του υπολογιστή. Όταν λαμβάνουμε το μήνυμα MV [A..H][1..8], σημαίνει ότι ο αντίπαλος έπαιξε κανονικά συνεπώς έχει έρθει η σειρά του AVR. Αφού ενημερώσουμε τον χάρτη με την κίνηση που έκανε ο αντίπαλος ξεκινάει ο υπολογισμός των κινήσεων του AVR. Οπότε και πάλι καλείται η συνάρτηση avrplaying και παίζει ο AVR. Όλα αυτά φυσικά εάν η κίνηση του αντιπάλου ήταν νόμιμη και εντός χρόνου. Σε διαφορετική περίπτωση στέλνουμε τα αντίστοιχα μηνύματα για timeout/illegal move.

Χρονομέτρηση αντιπάλου

Η χρονομέτρηση ξεκινάει όταν λάβουμε OK. Πρακτικά το OK εξυπηρετεί την σωστή συνομιλία μεταξύ υπολογιστή και AVR. Άρα μόλις λάβουμε το OK ξέρουμε ότι το μήνυμα ήταν σωστό και μεταδόθηκε χωρίς απώλειες. Από εκείνη τη στιγμή καλούμε την συνάρτηση που αρχικοποιεί τον μετρητή και από τότε ξεκινάει το μέτρημα του χρόνου για τον

αντίπαλο. Σε αυτό τον χρόνο κάνουμε την ενημέρωση του χάρτη με την κίνηση που κάναμε, εφόσον έγινε αυτή, και έπειτα περιμένουμε τον αντίπαλο.

Ο έλεγχος χρόνου για τον αντίπαλο πραγματοποιείται μόλις λάβουμε μήνυμα. Εάν έχουμε υπερβεί τον χρόνο μόλις μας ήλθε το μήνυμα, τότε δεν το λαμβάνουμε υπόψη και στέλνουμε μήνυμα ότι η κίνηση είναι εκτός χρόνου. Από την άλλη εάν είναι εντός χρόνου τότε ενεργοποιείται η διαδικασία αποκωδικοποίησης της εντολής, ελέγχου νόμιμης κίνησης αλλά και ενημέρωσης της σκακιέρας.

Όπως προ είπαμε το ΟΚ εξυπηρετεί την ολοκλήρωση της συνομιλίας μεταξύ υπολογιστή και AVR. Συνεπώς μόλις λάβουμε ΟΚ από το PC ο AVR ανάλογα το εκάστοτε flag της προηγούμενης κίνησης το οποίο έχει ενεργοποιηθεί ξέρει σε τι αναφέρεται το συγκεκριμένο ΟΚ. Για παράδειγμα σε περίπτωση IT ή IL ο PC μπορεί να πατήσει PL και σε αυτήν την περίπτωση ξανά ξεκινάει ο χρόνος για τον PC ο οποίος ξαναπαίζει. Σε αντίθετη περίπτωση όμως κατά την οποία στείλει ΟΚ τα ανάλογα flag θα δείξουν στον AVR ότι ο PC παραιτείται και ως αποτέλεσμα ο AVR νικάει. Τέλος το ΟΚ με ενεργοποιημένα τα αντίστοιχα flag, τα οποία υποδηλώνουν σωστή κίνηση είτε PS από τον AVR, γίνεται η ενημέρωση του χάρτη ή απλά εκκινεί ο χρόνος για τον αντίπαλο αντίστοιχα.

Το παιχνίδι θα τελειώσει είτε όταν γεμίσει ο πίνακας είτε εάν παραιτηθεί ο AVR.Για την περίπτωση κατά την οποία παραιτείται ο AVR θα εξηγηθεί αναλυτικά στο κομμάτι της στρατηγικής. Στην περίπτωση ολοκλήρωσης των κινήσεων και γεμίσματος του χάρτη η συνάρτηση win θα αναδείξει τον νικητή και θα ενεργοποιήσει το αντίστοιχο LED.

Χρονομέτρηση AVR

Η χρονομέτρηση του AVR εξασφαλίζει ότι η διαδικασία εύρεσης κίνησης δεν θα ξεπεράσει τον επιτρεπτό χρόνο παιχνιδιού. Αυτό σημαίνει ότι εάν η διαδικασία εύρεσης δεν έχει τερματίσει φτάνοντας το τέλος χρόνου τότε διακόπτεται και παίζεται η μέχρι τότε βέλτιστη κίνηση η οποία έχει βρεθεί. Ωστόσο παρόλο που τοποθετήθηκε η μέτρηση του AVR, η συνθήκη να ξεπεράσει τον επιτρεπτό χρόνο είναι πολύ δύσκολο καθώς χρησιμοποιώντας τον κρύσταλλο οποίος μας δόθηκε στα 10MHz η διαδικασία εύρεσης κίνησης με depth ίσο με δύο στην στρατηγική μας γίνεται σε λιγότερο από δύο second.

Στρατηγική απόφασης κίνησης AVR

Σε αυτό το σημείο είχαμε να επιλέξουμε ανάμεσα σε 3 στρατηγικές:

1. Min-max
2. Alpha-Beta Pruning
3. Monte Carlo

Ο min-max δημιουργεί ένα δένδρο με όλες τις πιθανές κινήσεις για το βάθος που του δίνεται σαν όρισμα. Μόλις φτάσει στο βάθος αυτό κάνει ένα evaluation του κατά πόσο είναι καλή ή όχι ή κίνηση και επιστρέφει αυτό τον αριθμό. Ο αλγόριθμος αυτός έχει δύο πλευρές: α) να μεγιστοποιήσει το αποτέλεσμα όταν κάνει κίνηση ο AVR, β) να ελαχιστοποιήσει το αποτέλεσμα όταν κάνει κίνηση ο PC. Πιο αναλυτικά:

Η συνάρτηση καλείται με όρισμα μία κίνηση, το χρώμα της κίνησης και το χάρτη πριν γίνει η κίνηση αυτή. Η συνάρτηση είναι αναδρομική και καλείται ανανεώνοντας τον χάρτη με την κίνηση που μόλις δόθηκε και με την κίνηση του αντιπάλου. Η διαφορά είναι

ότι όταν κάνει κίνηση ο AVR θέλει το αποτέλεσμα να είναι το καλύτερο δυνατό, αλλά όταν κάνει κίνηση ο PC θέλει να ελαχιστοποιηθεί αυτό όσο περισσότερο γίνεται. Οπότε η κλήση της συνάρτησης έχει και έναν Boolean ο οποίος καθορίζει κατά πόσο βρισκόμαστε στην πλευρά ελαχιστοποίησης του evaluation ή της μεγιστοποίησης αυτού. Όπως αναφέραμε και παραπάνω ο αλγόριθμος αυτός τρέχει όλες τις πιθανές κινήσεις, δηλαδή όλο το δέντρο με τρόπο breadth-first-search.

Εάν κάποιος κάνει μια μελέτη πάνω στον αλγόριθμο θα προσέξει πως κάποιες κινήσεις γίνονται χωρίς κάποιον ιδιαίτερο λόγο αφού θα μπορούσαν να παραληφθούν. Συγκεκριμένα εάν είμαστε στο σημείο της μεγιστοποίησης και μας έχει έρθει ένα evaluation με αριθμό 7 και στην επόμενη κίνηση ο αριθμός αυτός είναι μικρότερος του 7 αντιλαμβανόμαστε ότι δεν υπάρχει ουσία στο να συνεχίσει στην άλλη πλευρά του δέντρου αφού υπάρχει ήδη κάτι καλύτερο από αυτό που μόλις βρέθηκε. Αντίστοιχα και στην πλευρά που επιδιώκει να ελαχιστοποιηθεί το κέρδος του AVR. Αυτή ουσιαστικά η βελτίωση στον αλγόριθμο είναι ο alpha-beta pruning ο οποίος ονομάστηκε έτσι διότι κρατά δύο αριθμούς σε κάθε επίπεδο και τους συγκρίνει έτσι ώστε να αποφύγει ανούσιες επαναλήψεις της συνάρτησης.

Από την άλλη ο Monte-Carlo για όλες τις κινήσεις του AVR τερματίζει το παιχνίδι, αλλά και με όλες τις υποπεριπτώσεις που ακολουθούν, κρατώντας στατιστικά από το ποια κίνηση κέρδισε τα περισσότερα παιχνίδια αλλά και με την απόδοση που κέρδισε.

Κληθήκαμε σε αυτό το σημείο να αποφασίσουμε ποια από όλες αυτές τις στρατηγικές θα επιλέξουμε. Αναλογιζόμενοι ότι είμαστε σε ένα περιβάλλον με συγκεκριμένα και λιγοστά resources από άποψη μνήμης αλλά και η επιλογή να είναι ένας αποδοτικός και εύχρηστος αλγόριθμος. Με τη βοήθεια και ενός paper που βρήκαμε στο διαδίκτυο (reference υπάρχουν στη βιβλιογραφία) αλλά και των παραπάνω δεδομένων αποφασίσαμε να επιλέξουμε τον alpha-beta pruning. Ο min-max είναι αρκετά κακός στη απόδοση αφού κάνει κινήσεις που θα έπρεπε να παραλείπονται και από την άλλη ο Monte Carlo θα ήταν αδύνατον να ενσωματωθεί και να υλοποιηθεί σε μία μνήμη 1kb όσο καλή χρήση και διαχείριση να κάναμε.

Έτσι λοιπόν δημιουργήσαμε μία συνάρτηση η οποία έχει τα εξής ορίσματα:

1. Γραμμή κίνησης (A-H)
2. Στήλη κίνησης (1-8)
3. Βάθος σε κινήσεις
4. Alpha
5. Beta
6. Maximizing player (Boolean: true->max|false->min)
7. Χάρτης παιχνιδιού
8. Αριθμός από τα πούλια τα οποία έχουν τοποθετηθεί στον χάρτη.

Αρχικά έχουμε την γραμμή και την στήλη της εκάστοτε κίνησης για την οποία ξανακαλείται η συνάρτηση μας και με βάση αυτή γίνονται τα flips σε ένα αντίγραφο από τον χάρτη του παιχνιδιού μας. Στη συνέχεια θα προχωρήσει σε επόμενη πιθανή κίνηση και ανάλογα την μεταβλητή Boolean ο αλγόριθμος θα τρέξει την αντίστοιχη πλευρά για max ή min. Επίσης ορίζεται το βάθος με το οποίο εννοούμε πόσες κινήσεις μπροστά θα τρέξει ο αλγόριθμός μας. Στην δικιά μας περίπτωση είναι default το depth ίσο με δύο καθώς για μεγαλύτερο βάθος γεμίζει η μνήμη του AVR. Δύο σημαντικά ορίσματα είναι ο Alpha και ο Beta. Ο Alpha αξιοποιείται στην πλευρά του max στην συνάρτηση. Την πρώτη φορά αρχικοποείται στο -65 με αποτέλεσμα το evaluation το οποίο θα προκύψει να είναι σίγουρα μεγαλύτερο. Στη συνέχεια σε κάθε επανάληψη της αναδρομής για την πλευρά του max ανανεώνεται εάν βρεθεί κάτι μεγαλύτερο. Σε συνδυασμό με το Beta το οποίο ανανεώνεται στην πλευρά του

min και καλείται πρώτη φορά με +65, γίνεται η σύγκριση των δύο αριθμών και προκύπτει το ποιες κινήσεις θα <κλαδευτούν>. Το -65 και το +65 χρησιμοποιήθηκε μετά από μελέτη διότι η πιο ακραία περίπτωση στον πίνακα μας είναι να έχουμε 63 μαύρα/άσπρα και ένα του αντίθετου χρώματος. Το evaluation και στις δύο περιπτώσεις θα είναι είτε -62 είτε +62 συνεπώς το -65 και το +65 για την εφαρμογή μας μπορεί να θεωρηθεί άπειρο. Από την άλλη ο χάρτης παιχνιδιού δίνεται σαν όρισμα, αφού προηγουμένως έχει ενημερωθεί, δηλαδή αντιγράφεται και δίνεται σαν όρισμα στην επόμενη κλήση της αναδρομής. Να αναφέρουμε σε αυτό το σημείο ότι ο πίνακας των πιθανών κινήσεων έγινε με struct και όχι σταθερά ένας global πίνακας. Αυτό μας βοήθησε στο σημείο της στρατηγικής μας καθώς σε κάθε αναδρομή θέλαμε να κρατιούνται οι προηγούμενες πιθανές κινήσεις και όχι να πανογράφονται εάν είχαμε global.Τέλος ο αριθμός από τα πούλια εξυπηρετεί τον τέλος της αναδρομής καθώς υπάρχουν λιγότερες διαθέσιμες. Επίσης αν φτάσει στο τέλος του παιχνιδιού μέσα στην συνάρτηση και με οποιαδήποτε κίνηση ο AVR έχει αρνητικό αποτέλεσμα(δηλαδή χάνει) ανοίγει ένα flag το οποίο ουσιαστικά δηλώνει ότι ο AVR παραιτείται και δεν έχει νόημα να συνεχίσει να παίζει.

Παράρτημα

Να αναφέρουμε ότι στην συνάρτηση main υπάρχουν σε σχόλια ο κώδικας για εκτύπωση του πίνακα μας και σε άλλα σημεία(πχ analyzing_command() και στην avrplaying()) η εκτύπωση των πιθανών κινήσεων και των δύο παικτών. Με αυτό τον τρόπο μπορεί ο οποιοσδήποτε πολύ εύκολα να κάνει την επαλήθευση της σωστής λειτουργίας του παιχνιδιού.

Βιβλιογραφία

Παραθέτουμε το link στο οποίο βρήκαμε το paper για τις στρατηγικές.

http://web.eecs.utk.edu/~zzhang61/docs/reports/2014.04%20-%20Searching%20Algorithms%20in%20Playing%20Othello.pdf